Poste
1301

Postel
Recd
8Feb73

Network Working Group
Request For Comments #426
NIC #13011
Categories: Protocols, TELNET
References: 36,318,333,435

Bob Thomas, BBN-TENEX
January 26,1973

Reconnection Protocol

There are situations in which it is desirable to move one or both ends of a communication path from one host to another. This note describes several situations in which the ability to reconnect is useful, presents a mechanism to achieve reconnection, sketches how the mechanism could be added to Host-Host or TELNET protocol, and recommends a place for the mechanism in the protocol hierarchy.

1. Some Examples:

A. Consider the case of an executive program which TIP users could use to get network status information, send messages, link to other users, etc. Due to the TIP's limited resources the executive program would probably not run on the TIP itself but rather would run on one or more larger hosts who would be willing to share some of their resources with the TIP (see Figure 1).

The TIP user could access the executive by typing a command such as "@ EXEC"; the TIP would then ICP to Host1's executive port. After obtaining the latest network news and perhaps sending a few messages, the user

would be ready to log into Host2 (in general not the same as Host1) and do some work. At that point he would like to tell the executive program that he is ready to use Host2 and have the executive hand him off to Host2. To do this the executive program would first interact with Host2, telling it to expect a call from the TIP, and then would instruct the TIP to reconnect to Host2. When the user logs off Host2 he could be passed back to the executive at Host1 prepatory to doing more work elsewhere. The reconnection activity would be invisible to the TIP user.

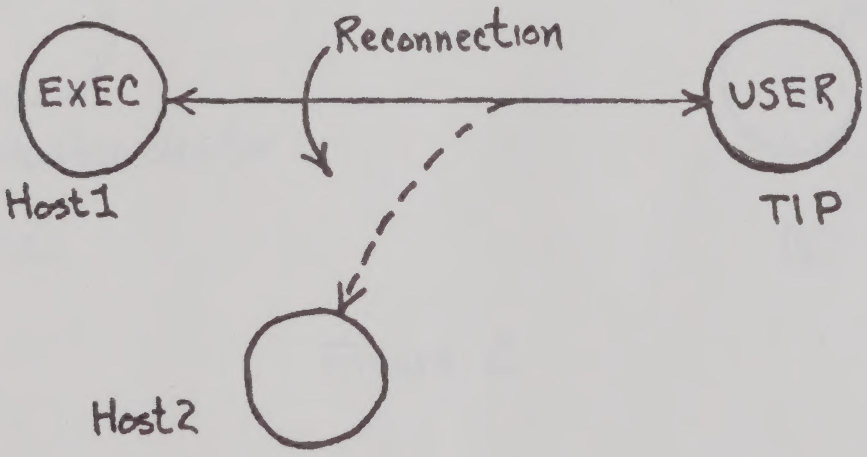


Figure 1

B. Imagine a scenario in which a user could use the same name and password (and perhaps account) to log into any server on the network. For reasons of security and economy it would be undesirable to have every name and password stored at every site. A user wanting to use a Host that doesn't have his name or password locally would connect to it and attempt to log in as usual (See Figure 2). The

Host, discovering that it doesn't know the user, would hand him off to a network authentication service which can determine whether the user is who he claims to be. If the user passes the authentication test he can be handed back to Host which can then provide him service*. The idea is that the shuffling of the user back and forth between Host and Authenticator should invisible to the the user.

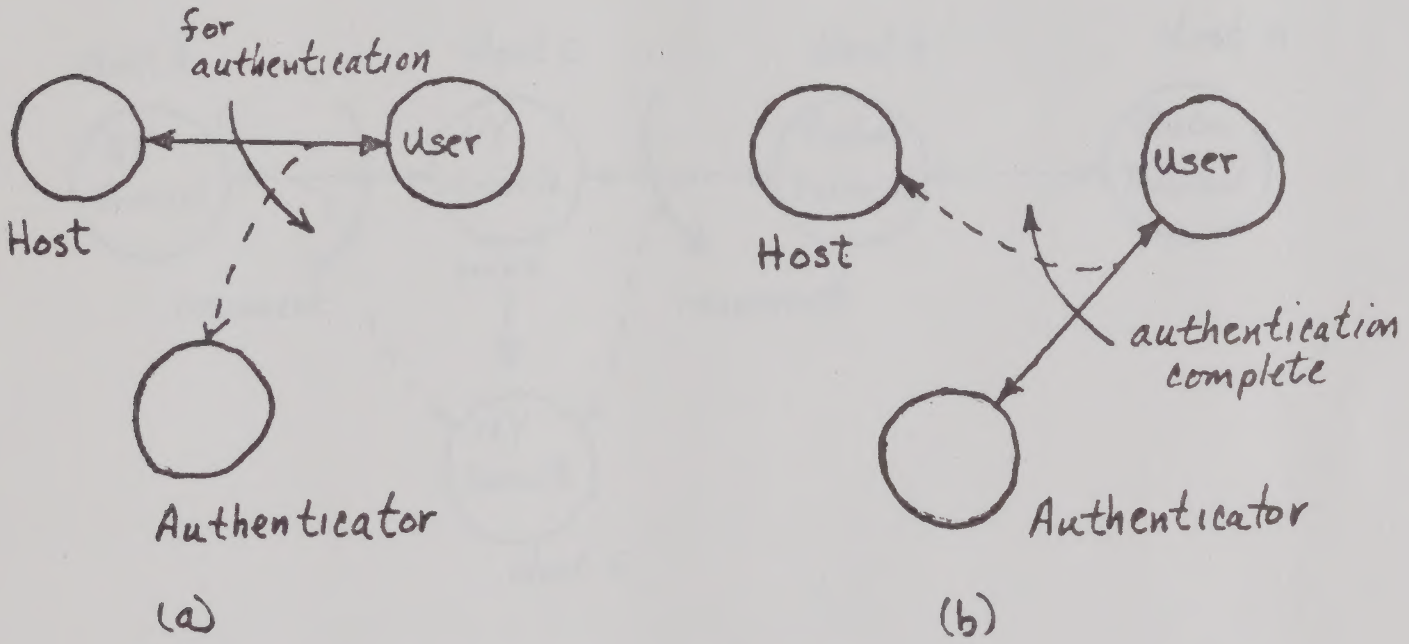

for authentication
User
Host
Authenticator
(a)
User
Host
authentication complete
Authenticator
(b)

Figure 2

C. The McROSS air traffic simulation system (see 1972 SJCC paper) already supports reconnection. It permits an on-going simulation to reconfigure itself by allowing parts to move from computer to computer. For example, in

---

* If the user doesn't trust the Host and is afraid that it might read his password rather than pass him off to the authenticator he could connect directly to the authentication service. After authentication, the Authenticator can pass him off to the Host.

a simulation of air traffic in the Northeast the program fragment simulating the New York Enroute air space could move from Host2 to Host5 (see Figure 3). As part of the reconfiguration process the New York Terminal area simulator and Boston Enroute area simulators break their connections with the New York Enroute simulator at Host2 and reconnect to it at Host5.

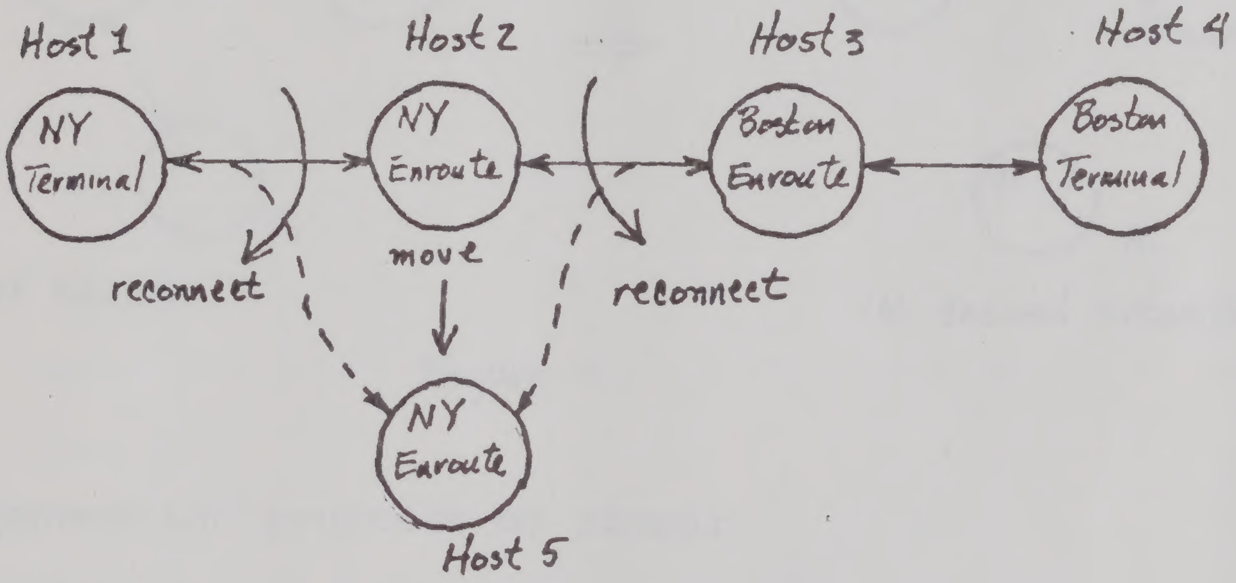

Host 1
Host 2
Host 3
Host 4
NY Terminal
NY Enroute
Boston Enroute
Boston Terminal
reconnect
move
reconnect
NY Enroute
Host 5

Figure 3

2. A Reconnection Mechanism

The mechanism proposed here could be added to the existing Host-Host protocol or to the TELNET protocol. The mechanism is first described and then its adaptation to each of the protocols is discussed.

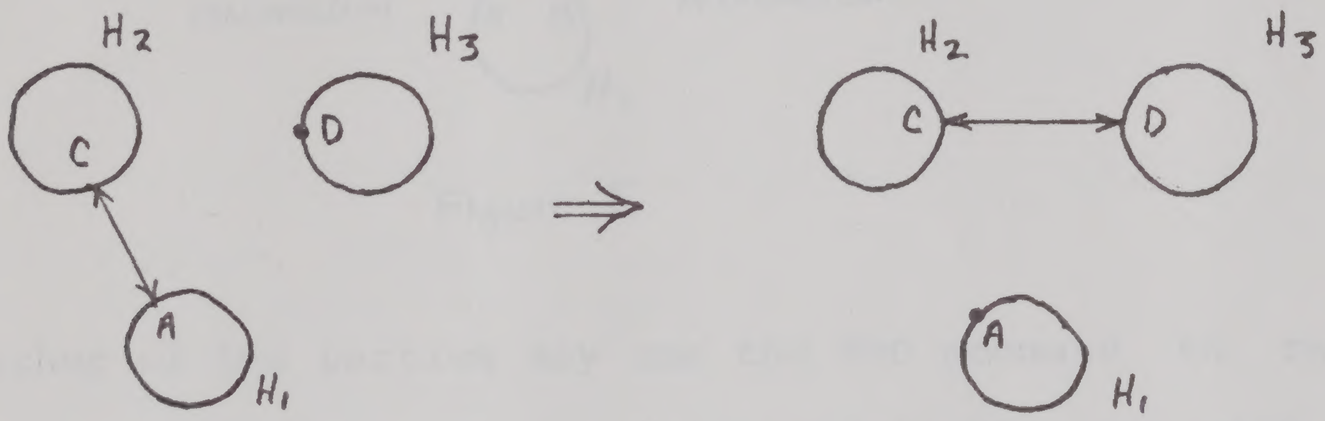
The reconnection mechanism includes four commands:

Reconnect Request: RRQ <path>
Reconnect OK: ROK <path>
Reconnect No: RNO <path>

Reconnect Do: RDO <path> <new destination>

where <path> is the communication path to be redirected to <new destination>.

Assume that H1 wants to move its end of communication path A-C from itself to port D at H3 (See figure 4).



(a) situation (b) desired situation

Figure 4

The reconnection proceeds by steps:

a. H1 arranges for the reconnection by sending RRQ to H2:
H1->H2: RRQ (path A-C)

b. H2 agrees to reconnect and acknowledges with ROK:
H2->H1: ROK (path C-A)

c. H1 notes that H2 has agreed to reconnect and instructs H2 to perform the reconnection:
H1->H2: RDO (path A-C) (Host H3,Port D)

d. H1 breaks paths A-C.
H2 breaks path C-A and initiates path C-D.

In order for the reconnection to succeed H1 must, of course, have arranged for H3's cooperation. One way H1 could do this would be to establish the path B-D and then proceed through the reconnection protocol exchange with H3 concurrently with its exchange with H2 (See Figure 5):

```
H1->H3:  RRQ (path B-D)
H3->H1:  ROK (path D-B)
H1->H3:  RDO (path B-D) (Host H2,Port C)
```

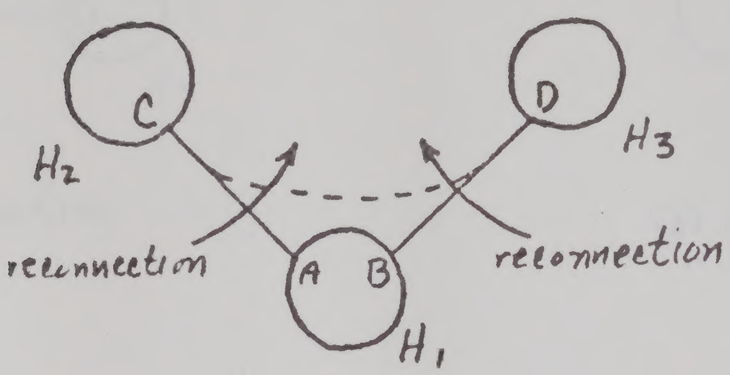
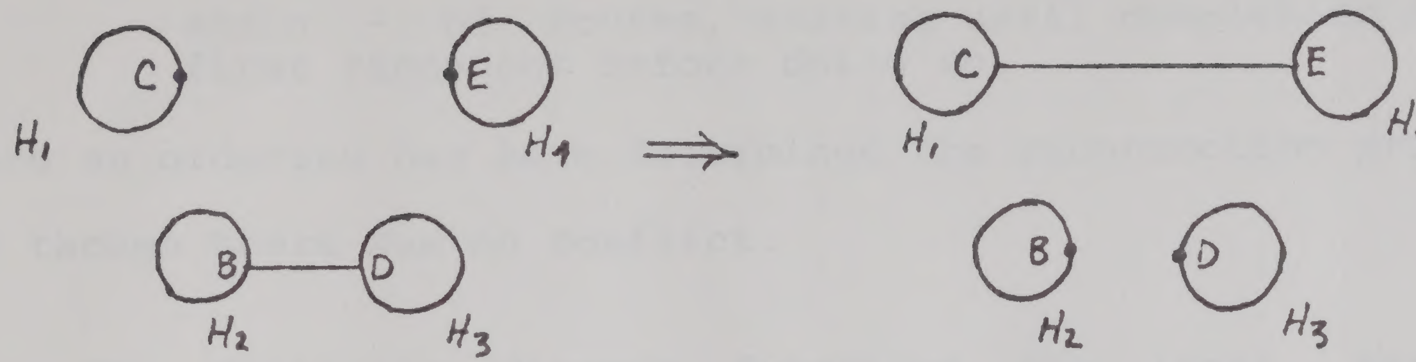


Figure 5

Either of the parties may use the RNO command to refuse or abort reconnection. H2 could respond to H1's RRQ with RNO; H1 can abort the reconnection by responding to ROK with RNO rather than RDO.

It is easy to insure that messages in transit are not lost during the reconnection. Receipt of the ROK message by H1 is taken to mean that no further messages are coming from H2; similarly receipt of RDO from H1 by H2 is taken to mean that no further messages are coming from H1.

To complete the specification of the reconnection mechanism consider the situation in which two "adjacent" entities initiate reconnections:



(a) situation

(b) desired situation

H2 and H3 "simultaneously" try to arrange for reconnection:

```
H2->H3:  RRQ (path B-D)
H3->H2:  RRQ (path D-B)
```

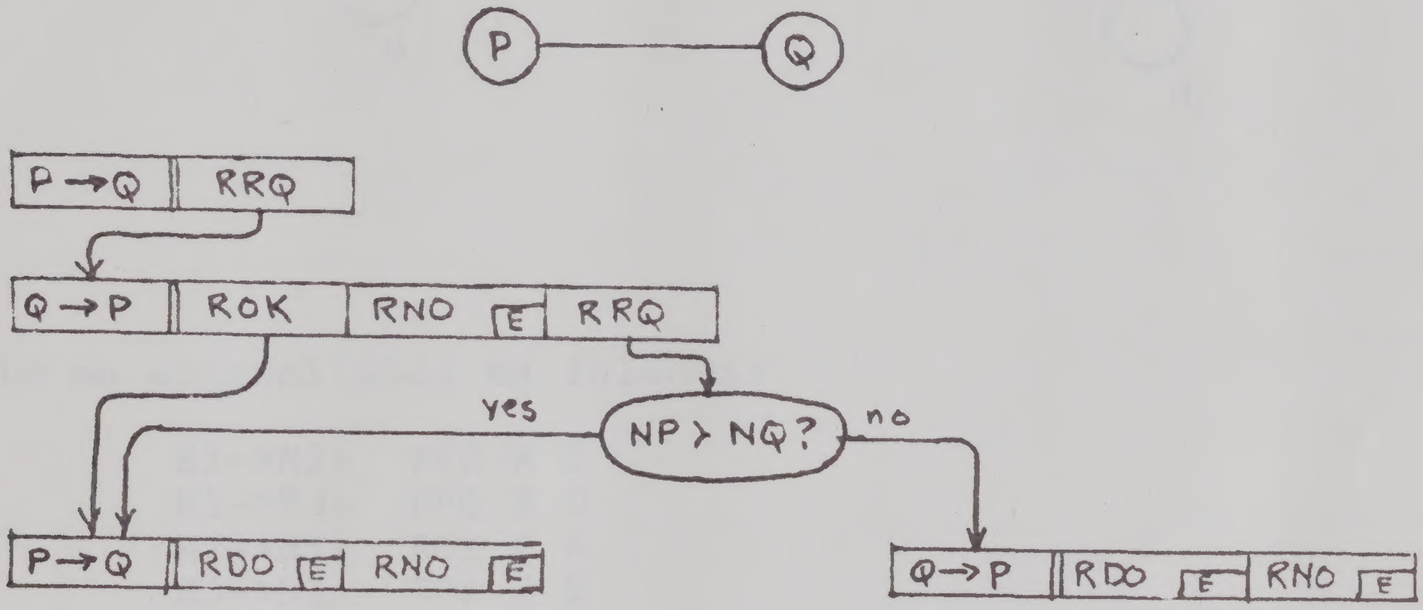
Thus, H2 sees an RRQ from H3 rather than an ROK or RNO in response to its RRQ to H3. This "race" situation can be resolved by having the reconnections proceed in series rather than in parallel: first one entity (say H2) performs its reconnect and then the other (H3) performs its reconnect. There are several means that could be used to decide which gets to go first. Perhaps the simplest is to base the decision on sockets and site addresses: the entity for which the 40 bit number formed by concatenating the 32 bit socket number with the 8 bit site address is largest gets to go first. Using this mechanism the rule is the following:

> If H2 receives an RRQ from H3 in response to an RRQ of its own:
>
> (let NH2,NH3 = the 40 bit numbers corresponding to H2 and H2)
>
> a. if NH2>NH3 then both H2 and H3 interpret H3's RRQ as an ROK in response to H2's RRQ.
>
> b. if NH2<NH3 then both interpret H3's RRQ as an RNO in response to H2's RRQ. This would be the only case in which it makes sense to "ignore" the refusal and try

again - of course, waiting until completion of the first reconnect before doing so.

Once an ordering has been determined the reconnection proceeds as though there was no conflict.

The following diagram describes the legal protocol command/response exchange sequences for a reconnection initiated by P:



NP and NQ are the 40 bit numbers for P and Q; $\boxed{E}$ indicates end of sequence.

3. Adding the Reconnection Mechanism to Host-Host Protocol

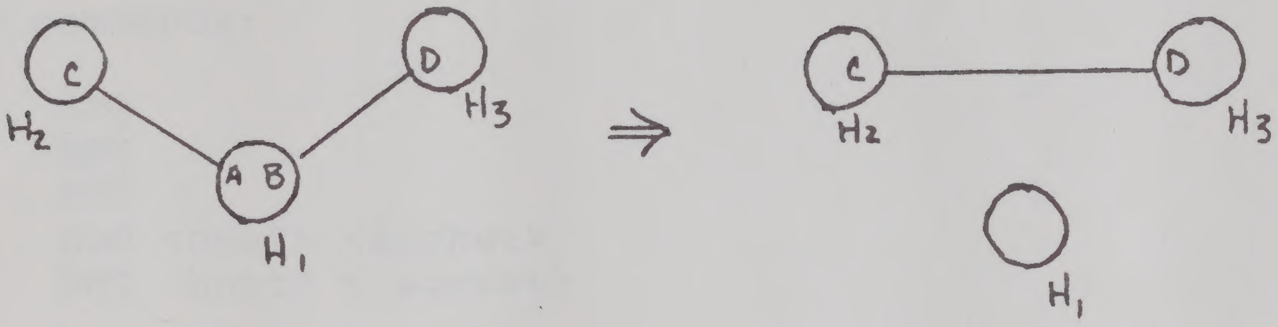
The four reconnect commands could be included directly in Host-Host protocol as follows:

RRQ <my socket> <your socket>
ROK <my socket> <your socket>
RNO <my socket> <your socket>
RDO <my socket> <your socket> <new host> <new socket>

The ROK and RDO commands for a send connection should not be

sent until there are no messages in transit over the connection. The RDO command is to be interpreted as a CLS.

The reconnection:



could be accomplished as follows:

```
H1->H2:  RRQ A C
H1->H3:  RRQ B D
H2->H1:  ROK C A
H3->H1:  ROK D B
H1->H2:  RDO A C H3 D
H1->H3:  RDO B D H2 C
H2->H1:  CLS C A
H3->H1:  CLS D B
H2->H3:  STR C D size
H3->H2:  RTS D C link
```

Note that it is possible for the STR from H2 to arrive at H3 before the RDO from H1. H3 must be prepared to queue such an RFC until it gets an RDO or RNO from H1. Stated differently, transmission of an ROK for a local socket causes the socket to move from an "open" state to a "reconnect pending" state and indicates willingness to queue subsequent RFC's until receipt of a "matching" RDO or RNO.

4. Reconnection in TELNET Protocol

Independently of whether Host-Host protocol directly supports reconnection, the reconnection mechanism could be included in TELNET with the addition to the TELNET protocol of the five commands:

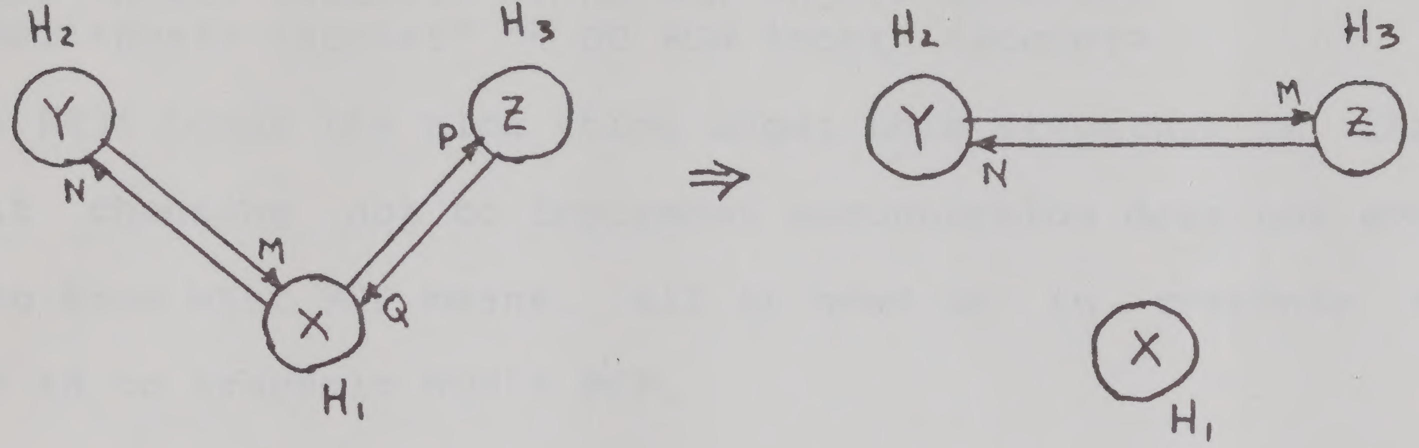
```
RRQ
ROK
RNO
RDO <host> <socket>
RWT <host> < socket>
```

where RRQ, ROK, RNO, RDO and RWT are appropriately chosen characters in the range 128 to 255. The first three commands require no parameters since they refer to the connections they are received on. For RDO and RWT, <host> is an 8 bit (= 1 TELNET character) host address and <socket> is a 32 bit (= 4 TELNET characters) number that specifies a TELNET receive socket at the specified host.

A pending reconnection can be activated with either RDO or RWT. The response to either is to first break the TELNET connection with the sender and then reopen the TELNET connection to the host and sockets specified. For RDO, the connection is to be reopened by sending two RFC's; for RWT, by waiting for two RFC's.

The RWT command is introduced to avoid races such as the one between the STR and CLS (RDO) discussed above. In Host-Host protocol the race is avoided by putting a connection into "reconnect pending" state upon transmission of ROK. For

TELNET the race can be avoided by the initiator of the reconnection by judicious use of RWT and RDO. For example the reconnection:



could be accomplished as follows:

X->Y: RRQ
X->Z: RRQ
Y->X: ROK
Z->X: ROK
X->Y: RWT H3 P
X closes connections to Y
Y closes connections to X
Y waits for STR and RTS from H3
x->Z: RDO H2 N
X closes connections to Z
Z closes connections to X
Z sends STR and RTS to H2 which Y answers with
  matching RTS and STR to compete reconnection

The reconnect mechanism for TELNET can be made to fit nicely into the command format suggested by Cosell and Walden in RFC #435. Consider the addition of three new commands to TELNET:

Prepare for Reconnect:	RCP
Begin Reconnect by sending RFC's:	RCS
Begin Reconnect by waiting for RFC's:	RCW

Using these three command and the DO, DON'T, WILL, WON'T

commands of RFC #435, the commands proposed earlier become:

```
RRQ => DO RCP
ROK => WILL RCP
RNO => WON'T RCP ;for responses to DO RCP
       DON'T RCP ;for responses to WILL RCP
                 ;i.e. used to cancel an RCP.
RDO <host> <socket> => DO RCS <host> <socket>
RWT <host> <socket> => DO RCW <host> <socket>
```

As RFC #435 notes the nice thing about this structure is that a host choosing not to implement reconnection does not even have to know what RCP means. All it need do in response to DO RCP is to transmit WON'T RCP.

5. Recommendation

I feel that reconnection is a basic notion and that its proper place within the protocol hierarchy is at the Host-Host level where it would be available for use in all higher level protocols. The difficulty is that placing it there would, of course, require NCP rewrites. Reluctance to make NCP modifications would probably be sufficient to kill interest in the proposal.

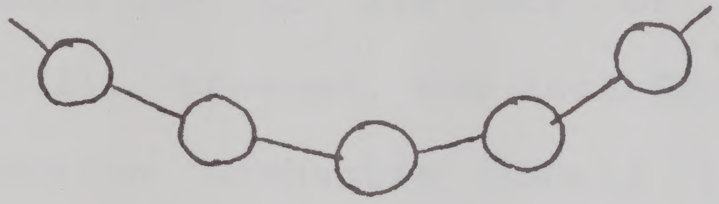
Therefore, for pragmatic reasons, I recommend that the reconnection mechanism be included in TELNET as an "option" in the spirit of RFC #435. This can be accomplished with the addition to the TELNET protocol of the RCP, RCS, RCW commands as described in Section 4. Modification of user- and server-TELNET programs to handle these new commands should be straightforward. If a reconnection option is made part of

TELNET protocol the TENEX hosts will support it. In addition, the TIP guys (Walden and Cosell) have said that they like the reconnection mechanism and have agreed, in principle, to implement it for TIPs if it is accepted as part of TELNET protocol.

Addition of reconnection at the TELNET level rather than the Host-Host level is admittedly a compromise. However, with it, activity of the sort described in Examples A and B of Section 1 will be possible.

6. Additional Remarks

A. Reconnection is not a new notion. An early proposal for Host-Host protocol (RFC #36) included facilities to support reconnection. The reconnection mechanism in RFC #36 supposes a configuration in which entities are "daisy-chained" together by connections:



and specifies how one or more entities can break out of the chain. As suggested above (Figure 5) the mechanism proposed in this note supports that kind of reconnection.

B. In practice one would expect simultaneous initiation of reconnects by adjacent entities to be relatively rare.

C. The approach taken in RFC #36 to handle simultaneous reconnection attempts by entities adjacent in the chain is to accomplish the reconnect as a single, carefully coordinated, global reconnect. I feel that the sequence of locally coordinated reconnects as proposed above is preferable. When N adjacent entities simultaneously attempt reconnection the single, globally coordinated reconnect as outlined in RFC #36 requires ∿ N^2 control messages whereas the sequential locally coordinated reconnect requires ∿ N.

D. A word about security is in order. It should be clear that the decision to accept or reject a particular reconnection request is the responsibility of the entity (person at a terminal or process) using the connection. In many cases the entity may choose to delegate that responsibility to its NCP or TELNET (e.g., Example A, Section 1). However, the interface a Host provides to the reconnection mechanism should include means for local entities to exercise control over response to remotely initiated reconnection requests. For example, a user-TELNET might support several modes of operation with respect to remotely initiated reconnections:

1. transparent: all requested reconnections are to be performed in a way that is invisible to the user;
2. visible: all requested reconnections are to be performed and the user is to be informed whenever a reconnection occurs;
3. confirmation: the user is to be informed of each reconnection request which he may accept or reject;
4. rejection: all requested reconnects are to be rejected.

E. Reconnection can be achieved almost trivially within the Message Switched Protocol (MSP) proposed by Bressler, Murphy and Walden in RFC #333 (within MSP, "reconnection" is probably not the correct term). For example use of the following conventions with that MSP (expressed in the terminology of RFC #333) support reconnection:

1. unless a reconnection is in progress, rendezvous is to occur at the sending site;

2. the receiving end of a communication path can be moved by passing the names of the rendezvous site and the ports to the new receiver;
3. receipt of an OUT message for which the source site differs from the rendezvous site signals that the sending end is being moved; the source site should be used as the rendezvous site for subsequent IN messages;
4. the sending end of a communication path can be moved by passing the names of the ports to the new sender; to complete the move the new sender uses the previous sender's site as rendezvous site for its first OUT message and its own site as rendezvous for subsequent OUT nessages.

As simple and appealing as this procedure seems, I doubt that it would be used in practice if MSP were to be implemented as a replacement for or alternative to existing Host-Host protocol. The reason is that the ability to pass ports from Host to Host (needlessly) complicates port name allocation by requiring cooperation among Hosts to manage the allocation (e.g., before a Host can safely allocate a port name for use by a local process it must not only insure that the port is not in use locally but also that no process out in the network is using it.) The inter-Host cooperation required to support the passage of ports among Hosts can probably not be reliably achieved in practice. Therefore port passage of the sort described in RFC #333 should not be supported at the Host-Host protocol level. For this reason, I feel that within an MSP "reconnection" would be best handled by a mechanism such as the one proposed in this note.